

Applicant : Cadir Batista Lee, et al.  
Serial No. : 09/712,855  
Filed : November 14, 2000  
Page : 10



Attorney's Docket No.: 09595-004002

RECEIVED  
MAY 7 2001  
Group 2100

REMARKS

Applicant asks that all claims be examined. Enclosed is a check in the amount of \$1,000.00 for excess claim fees. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: \_\_\_\_\_

4-27-01

A handwritten signature in black ink, appearing to read "KK Lutton", written over a horizontal line.

Katherine Kelly Lutton  
Reg. No. 46,333

Fish & Richardson P.C.  
2200 Sand Hill Road, Suite 100  
Menlo Park, CA 94025  
Telephone: (650) 322-5070  
Facsimile: (650) 854-0875

**Version with markings to show changes made**

In the specification:

Paragraph beginning at page 6, line number 3, has been amended as follows:

Fig. 4 is a ~~process~~flowchart illustrating a process for comparing keys in Fig. 3.

Paragraph beginning at page 6, line number 26, has been amended as follows:

Fig. 1 shows a computer system 100 with one or more vaults ~~is shown~~. The system 100 has one or more clients 102, each of which has a set of catalogs 104, as well as a client vault 106. The client vault 106 is a computer readable medium which stores one or more software components (entities) which are designated by the set of catalogs 104. In this case, the client vault 106 exists on a data storage device such as a hard drive on the computer system 100. Alternatively, the vault may reside on one or more data storage devices connected to a network 110, as discussed below. Each of the software components may be referenced by more than one application, and the software components are used to reconstruct the application.

Paragraph beginning at page 9, line number 15, has been amended as follows:

Turning now to Fig. 3, a process 140 for accessing files stored on one of the vaults 130-132 is shown. The process 140 first generates a key from a metadata file (step 142). The metadata file identifies all elements that make up a single application, as identified using a state probe. The operation of the state probe is described in more detail in a ~~co-pending application~~ entitled Automatic Configuration Management, @ Application Serial No. 08/993,103, filed on December 18, 1997 U.S. Patent No. 5,996,073, entitled "System and Method for Determining Computer Application State," issued on November 30, 1999, the content of which is incorporated by reference.

Paragraph beginning at page 9, line number 30, has been amended as follows:

If the component files of the application software to be recreated using the key are large and therefore cumbersome to transfer, it is more efficient to determine whether the component files of the application software already exist locally. Such determination may be made by first looking up the key on the client 122 (step 144) and then optionally comparing the key generated from the metadata file to the key on the client 122 (step 146) and ~~is detailed in Fig. 3 below~~ described in reference to Fig. 4. The key comparison process sets a flag if a difference exists and otherwise clears the flags.

Paragraph beginning at page 11, line number 8, has been amended as follows:

The process of Fig. 4 checks whether the sequence attributes are equal (step 166). If so, the difference flag is set (step 168). From step ~~116~~166, if the attributes are not equal, the process checks whether one of the attributes is newer than the other (step 170). If so, the process proceeds to step 168 to set the difference flag. Alternatively, in the event that the attribute is not newer, the process then checks whether or not one of the attributes is older (step 172). If no, the process proceeds to step 168 to set the difference flag. Alternatively, in the event that one of the attributes is older, the process determines whether or not the file may be overwritten (step 174). If so, the difference flag is set (step 168). Alternatively, the process exits (step 176).

Paragraph beginning at page 11, line number 21, has been amended as follows:

Referring now to Fig. 5, a flowchart illustrating a post-process ~~170~~180 for placing files onto the vault is shown. The post-process 180 first generates a post key from the metadata file (step 182). Optionally, the process 180 may look up the key present on the vault (step 184) and compare the keys (step 186). If the comparison causes the difference flag to be set (step 188), the key from the metadata is used to post the file to the vault from the client (step 190). From step 188 or step 190, the process of Fig. 5 exits (step ~~192~~191). Pseudo-code for the process ~~170~~180 is as follows:

```
Transform metadata descriptor into key
for each vault
    directly access key on vault
    if found then end for
next
if not found then
    locate first writable vault
    insert file with key
end if
return
```

Paragraph beginning at page 12, line number 7, has been amended as follows:

Turning now to Fig. 6, the process 182 of Fig. 5 to generate the post key from the metadata file is shown in more detail. In Fig. 6, metadata associated with each file is generated (step ~~194~~183). Next, the process 182 verifies the integrity of the file (step ~~196~~185).

Paragraph beginning at page 12, line number 21, has been amended as follows:

A key is then generated from the metadata (step ~~199~~ 187) before the process 182 exits (step 189). Key generation should include an integrity checksum as described above as well as basic information about the size, name, and attributes of the file. In the form of a checksum, the key allows identity information as well as integrity information to be easily verified.

Paragraph beginning at page 13, line number 19, has been amended as follows:

Figs. 8A and 8B show alternate processes to provide state-based software life cycle management using a vault. Turning now to Fig. 8A, a process for performing software management first generates the metadata (step 403), as described in Fig. 1. ~~The metadata may include DNA information, as described in more detail in the incorporated by reference application.~~ The information is then used to maintain software (step 405) before the process exits. Correspondingly, Fig. 8B shows a second software life cycle management process. Initially, the metadata information is generated and published (step 410). Next, components of the software are replicated (step 450) based on the metadata. The software is then installed (step 470). After installation, the software may be maintained (step 490).

Paragraph beginning at page 13, line number 33, has been amended as follows:

Turning now to Fig. 9, the metadata publication step 410 is shown in more detail. In Fig. 9, a vault is located (step 412). The vault may be a server that maintains items referenced in the metadata files. Next, component files associated with the software are stored in the vault (step 414). Similarly, ~~metadata including DNA information is stored in the vault (step 416).~~ A catalog, or an index of metadata files that represent the physical components of the software being published, is updated (step 418). Finally, the process 410 exits (step 420).

Paragraph beginning at page 14, line number 30, has been amended as follows:

The replicate step 450 of Fig. 8B is shown in more detail in Fig. 11. First, the source vault is located (step 452). Next, the destination vault is located (step 454). Files are then transferred from the source vault to the destination vault (step ~~546~~ 456). Similarly, metadata information is copied from the source vault to the destination vault (step 458). Finally, the vault catalog is updated (step 460) before the process of Fig. 11 exits (step 462).

Paragraph beginning at page 15, line number 3, has been amended as follows:

Turning now to Fig. 12, the installation step 470 (Fig. 8B) is shown in more detail in Fig. 12. First, the vault catalog is loaded (step 472). Next, the highest version of the software stored

in the vault is determined (step 474). The metadata associated with the highest version of the software is copied to the target machine (step 476). Further, data is remapped (step 478). The process of Fig. 12 then applies a preprocessing operation (~~pre-RNA~~) to the remapped data (step 480) to convert data into the proper format and set up variables appropriately, among others. Further, items associated with the software are installed (step 482). A post-processing (~~post-RNA~~) process is applied (step 484). This step is similar to step 480 in that variables are checked and data is formatted. Finally, an inventory of the software being installed is updated (step 486) before the process exits (step 488).

Paragraph beginning at page 15, line number 18, has been amended as follows:

Turning now to Fig. 13, the maintenance step 490 of Fig. 8B is shown in detail. The maintenance step 490 is an event driven process and thus receives a software trigger event (step 492). Based on the trigger event, the process of Fig. 13 determines various possible events, including a check for update event 494, a protect software event (~~step 495~~ 496), a software recovery (step 498) event, a check removal event (step 500), and an examine system event (step 502). From steps 494-502, the triggering event is reported (step 504) before the process of Fig. 13 exits (step 506).